Forensic Acoustics: An Introduction to Voice Identification

Iván López-Espejo, PhD

EAA SUMMER SCHOOL
Applications of Sound Signal Processing in Society and Digital Industry

iloes@ugr.es

Saturday 21st June, 2025

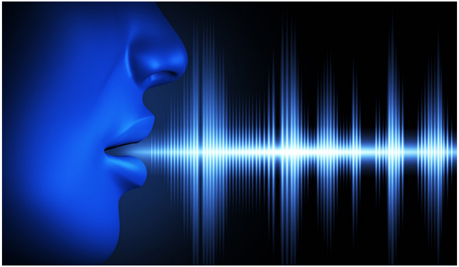
Overview

Overview

1. Introduction to Speaker Verification
2. Implementation of a Speaker Verification System

Introduction to Speaker Verification

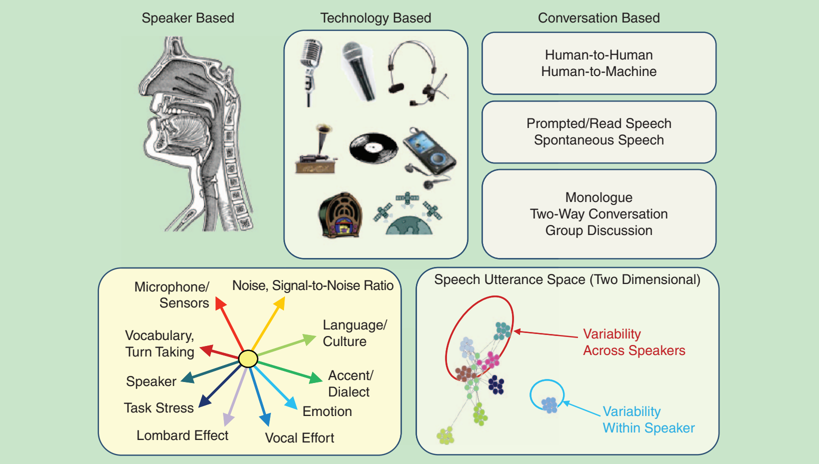
- **Speaker recognition** systems have become an important means of verifying identity in e-commerce applications, forensic science, law enforcement, etc.
- Two main approaches in voice biometrics:
  1. **Identification/Recognition** $(1:N)$: Closed scenario *vs.* open scenario
  2. **Verification** $(1:1)$
- Text-independence *vs.* text-dependence



[SoftReport] The Software Report, "Voice Recognition Technology Holds A Wealth Of Benefits For SaaS," `https://www.thesoftwarereport.com/voice-recognition-technology-holds-a-wealth-of-benefits-for-saas/`

Introduction to Speaker Verification

- Sources of variability in the context of speaker recognition:



[Hansen15] J. H. L. Hansen and T. Hasan, "Speaker Recognition by Machines and Humans: A tutorial review," IEEE Signal Processing Magazine, 2015

Introduction to Speaker Verification

- **Recursive enrollment**: We update a speaker's reference sample after a successful verification to strengthen the system against intra-speaker variability
  1. *Aging*
  2. *Disease*
  3. *Mood*
  4. ...

- **If and only if the verification was successful!** (i.e., $s' = s$):

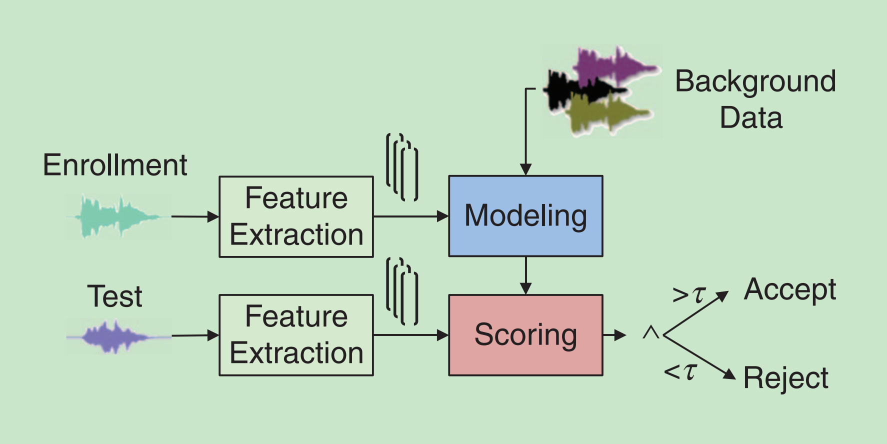
$$\mathbf{e}_t^{(s)} = \lambda(\sigma(x), \Psi, \gamma)\mathbf{e}_{t-1}^{(s)} + (1 - \lambda(\sigma(x), \Psi, \gamma))\mathbf{v}_t^{(s')}$$

  - $\mathbf{v}_t^{(s')}$: New verification sample from speaker s'
  - $\mathbf{e}_{t-1}^{(s)}$: Previous reference sample from speaker s
  - $\mathbf{e}_t^{(s)}$: Updated reference sample from speaker s
  - $\lambda(\sigma(x), \Psi, \gamma)$: Remembering factor dependent on the calibrated score $\sigma(x)$

[Espejo24] I. López-Espejo *et al.*, "Authenticating a User," US Patent, 2024

Introduction to Speaker Verification

- Block diagram of a **basic speaker verification system**:



[Hansen15] J. H. L. Hansen and T. Hasan, "Speaker Recognition by Machines and Humans: A tutorial review," IEEE Signal Processing Magazine, 2015

Databases

- VoxCeleb1: More than 100k utterances from 1,251 celebrities scraped from YouTube
- VoxCeleb2: More than 1M utterances from 6,112 celebrities scraped from YouTube

[Nagrani17] A. Nagrani *et al.*, "VoxCeleb: a large-scale speaker identification dataset," in Proc. of Interspeech 2017

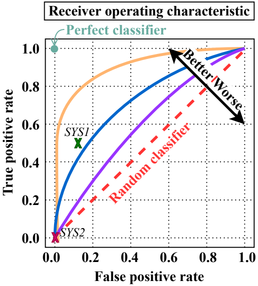
[Chung18] J. S. Chung *et al.*, "VoxCeleb2: Deep Speaker Recognition," in Proc. of Interspeech 2018

- **NIST SRE (Speaker Recognition Evaluation)**: From 1996 to date
- NIST SRE 2018

`https://sre.nist.gov/`

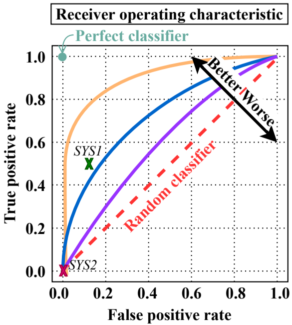
ROC and DET Curves

- Probability that a positive sample is correctly detected as such:
$$\text{True Positive Rate (TPR)} = \text{Recall} \equiv \frac{TP}{TP + FN}$$
- Probability that a negative sample is incorrectly classified as positive:
$$\text{False Positive Rate (FPR)} \equiv \frac{FP}{FP + TN}$$
- The **receiver operating characteristic (ROC)** curve is obtained by sweeping the sensitivity/decision threshold:



ROC and DET Curves

- **Area under the ROC curve ($\mathbf{AUC_{ROC}} \in [0, 1]$):** Probability that a classifier will rank a randomly chosen positive sample higher than a randomly chosen negative sample



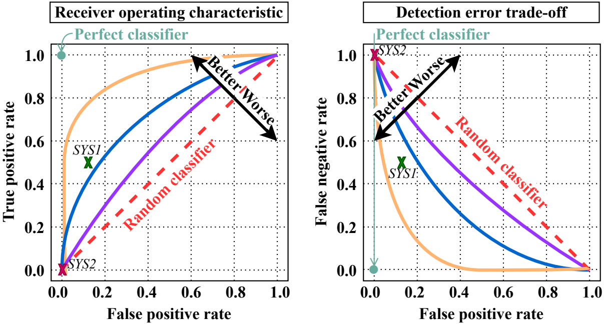
Ground truth	NK	NK	KW	NK	NK	KW	NK	NK	NK	NK
SYS1	NK	NK	KW	NK	NK	NK	KW	NK	NK	NK
SYS2	NK	NK	NK	NK	NK	NK	NK	NK	NK	NK

ROC and DET Curves

- Given that

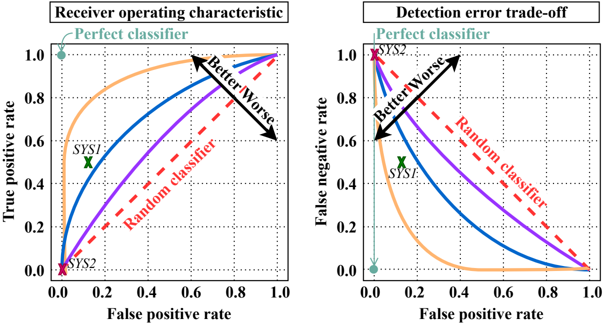
$$\text{False Negative Rate (FNR)} \equiv \frac{FN}{FN + TP} = 1 - \text{TPR},$$

the **detection error trade-off (DET)** curve is simply a vertically-flipped version of the ROC curve:

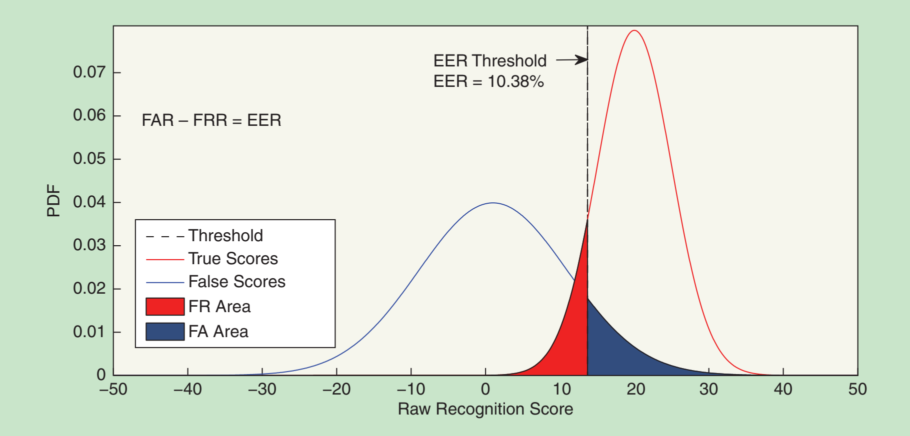


ROC and DET Curves

- **Area under the DET curve ($\mathbf{AUC_{DET}} \in [0,1]$):** The smaller, the better
- **Equal error rate (EER):** Intersection point between the identity function and the DET curve (i.e., the point where FNR = FPR)



Setting the Decision Threshold



[Hansen15] J. H. L. Hansen and T. Hasan, "Speaker Recognition by Machines and Humans: A tutorial review," IEEE Signal Processing Magazine, 2015

Detection Cost Function

- The detection cost function (DCF) was proposed by **NIST (National Institute of Standards and Technology)**

$$\mathrm{DCF}(\tau) = C_{miss} P_{miss}(\tau) P_{target} + C_{FA} P_{FA}(\tau)(1 - P_{target})$$

1. C_{miss}: Cost of a false negative (e.g., 10)
2. C_{FA}: Cost of a false positive (e.g., 1)
3. P_{target}: Prior probability of target speaker (e.g., 0.01)
4. $P_{miss}(\tau)$: Probability of a false negative given a threshold τ
5. $P_{FA}(\tau)$: Probability of a false positive given a threshold τ

- The goal would be to find the value of τ that minimizes $\mathrm{DCF}(\tau)$

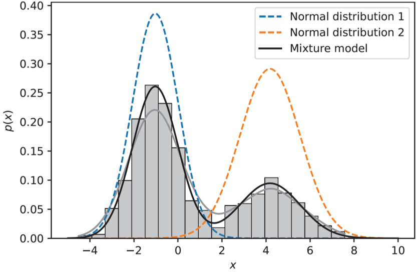
Introduction to Speaker Verification

FIVE MAIN GENERATIONS

1. **GMM-UBM (Gaussian Mixture Models-Universal Background Model)**: Technology based on Gaussian mixture models
2. **GMM-SVM (Gaussian Mixture Models-Support Vector Machines)**: GMM supervectors classified by SVMs
3. **JFA (Joint Factor Analysis)**: Decomposition of total variability into speaker and channel/session components
4. **i-vectors (identity vectors)**: Modeling of total variability
5. **Neural networks**: Technology based on deep learning

1st Generation: GMM-UBM (2000)

$\mu_1 = -1$, $\sigma_1 = 1$ — $\mu_2 = 4$, $\sigma_2 = 1.5$
$w_1 = 0.7$ — $w_2 = 0.3$



[Yehoshua23] R. Yehoshua, "Gaussian Mixture Models (GMMs): from Theory to Implementation," `https://towardsdatascience.com/gaussian-mixture-models-gmms-from-theory-to-implementation/`

Gaussian mixture models:

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}}{\sqrt{(2\pi)^d|\boldsymbol{\Sigma}|}}$$

$$p(\mathbf{x}) = \sum_{k=1}^{\mathcal{K}} w_k \mathcal{N}_k(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

$$\sum_{k=1}^{\mathcal{K}} w_k = 1$$

$$0 \leq w_k \leq 1 \quad \forall k = 1, ..., \mathcal{K}$$

$$\theta = \{w_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k;\ 1 \leq k \leq \mathcal{K}\}$$

Given a training dataset $\mathbf{X} = \left\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(N)}\right\}$,

$\theta^* = \arg\max_\theta p(\mathbf{X}|\theta) = \arg\max_\theta \prod_{i=1}^{N} p(\mathbf{x}^{(i)}|\theta)$ (Expectation-maximization!)

1st Generation: GMM-UBM (2000)

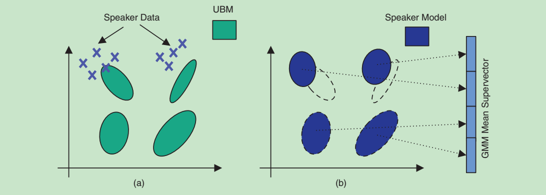
- λ_s: GMM for speaker s (MAP adaptation of a UBM)
- λ_0: UBM
- $\mathbf{X} = \{\mathbf{x}_n;\ n = 1, ..., T\}$ are feature vectors from an observation O

Hypothesis contrast:

1. H_0: O comes from speaker s
2. H_1: O does not come from speaker s

We calculate $\Lambda(\mathbf{X}) = \log\left(\frac{p(\mathbf{X}|\lambda_s)}{p(\mathbf{X}|\lambda_0)}\right) = \log p(\mathbf{X}|\lambda_s) - \log p(\mathbf{X}|\lambda_0)$

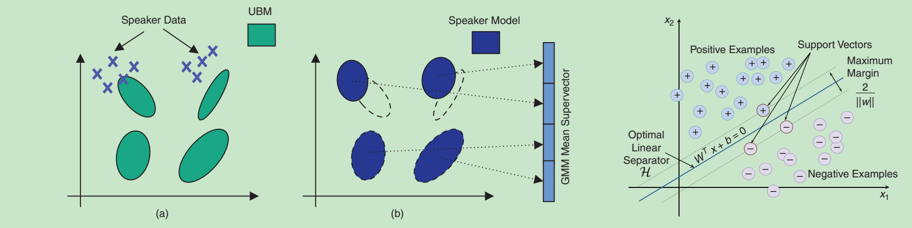
If $\Lambda(\mathbf{X}) \geq \tau$, we then accept the hypothesis H_0



[Hansen15] J. H. L. Hansen and T. Hasan, "Speaker Recognition by Machines and Humans: A tutorial review," IEEE Signal Processing Magazine, 2015

2nd Generation: GMM-SVM (2006)

- GMM supervectors provide speaker representations of a **fixed dimensionality**
- **GMM-SVM**: GMM supervector classification by means of SVMs (*Support Vector Machines*)



[Hansen15] J. H. L. Hansen and T. Hasan, "Speaker Recognition by Machines and Humans: A tutorial review," IEEE Signal Processing Magazine, 2015

3rd Generation: JFA (2004)

- **FA (Factor Analysis)**: Method for explaining speaker and channel variability in the **supervector** space

$\mathbf{m}_{s,h}$: GMM supervector for speaker s (and for session h):

$$\mathbf{m}_{s,h} = \mathbf{m}_0 + \mathbf{m}_{spk} + \mathbf{m}_{chn} + \mathbf{m}_{res}$$

1. $\mathbf{m}_0$: Environment-, channel-, and speaker-independent component (constant, from the UBM)
2. $\mathbf{m}_{spk}$: Speaker-dependent component
3. $\mathbf{m}_{chn}$: Environment- and channel-dependent component
4. $\mathbf{m}_{res}$: Residue

3rd Generation: JFA (2004)

- **Joint Factor Analysis – JFA** (*in the GMM supervector domain*):

$$\mathbf{m}_{s,h} = \mathbf{m}_0 + \underbrace{\mathbf{U}\mathbf{x}_h}_{\mathbf{m}_{chn}} + \underbrace{\mathbf{V}\mathbf{y}_s}_{\mathbf{m}_{spk}} + \underbrace{\mathbf{D}\mathbf{z}_{s,h}}_{\mathbf{m}_{res}}$$

- **U** and **V** are low-rank matrices estimated during a **training** phase by a PCA-like dimensionality reduction algorithm

- **D** is a diagonal matrix estimated along with **U** and **V** by an EM-like algorithm

- Given a supervector $\mathbf{m}_{s,h}$, $\mathbf{x}_h$, $\mathbf{y}_s$ and $\mathbf{z}_{s,h}$ (the **channel, speaker, and residual factors**) are obtained by MAP or Bayesian estimation taking into consideration that $(\mathbf{x}_h, \mathbf{y}_s, \mathbf{z}_{s,h}) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

- $\mathbf{y}_s$ is the speaker **feature vector** used **for comparison/verification**

4th Generation: i-vectors (2009)

- Dr. Najim Dehak researched the use of **JFA as a feature extractor** ($\mathbf{y}_s$) and **SVMs for classification**
- He realized that $\mathbf{x}_h$ *also comprised speaker-dependent information*
- *Total variability space*: He decided to combine speaker and channel factors into a single space: $\mathbf{m}_{s,h} = \mathbf{m}_0 + \mathbf{T}\mathbf{w}_{s,h}$

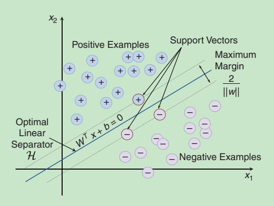
$\mathbf{T}$ is the total variability low-rank matrix and $\mathbf{w}_{s,h} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

i-vector: $\mathbf{w}^*_{s,h} = E[\mathbf{w}_{s,h}|\mathbf{F}]$

$\mathbf{F} = \sum_n \gamma_k(n) \left(\mathbf{x}_n - \boldsymbol{\mu}_k^{UBM}\right)$

$$\gamma_k(n) = P(k|\mathbf{x}_n) = \frac{p(\mathbf{x}_n|k)P(k)}{p(\mathbf{x}_n)} = \frac{\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)w_k}{\sum_k w_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}$$

Najim Dehak, PhD

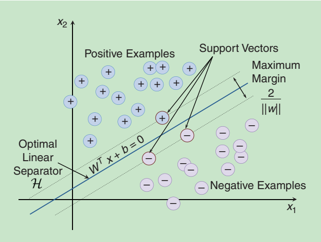


[Hansen15] J. H. L. Hansen and T. Hasan, “Speaker Recognition by Machines and Humans: A tutorial review,” IEEE Signal Processing Magazine, 2015

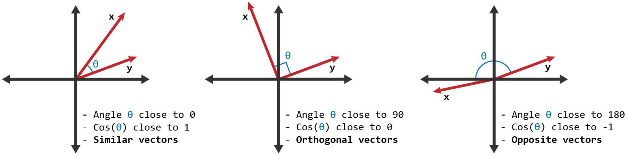
But... How do we Use $\mathbf{y}_s$ and $\mathbf{w}_{s,h}$ for Verification?

- We extract $\mathbf{w}_{test}$ from a sample for verification and claim the identity associated with $\mathbf{w}_{target}$

- **SVMs (Support Vector Machines)**



[Hansen15] J. H. L. Hansen and T. Hasan, "Speaker Recognition by Machines and Humans: A tutorial review," IEEE Signal Processing Magazine, 2015



[Karabiber24] F. Karabiber, "Cosine Similarity," `https://www.learndatasci.com/glossary/cosine-similarity/`

- **Cosine similarity**:

$$\mathcal{S}_c(\mathbf{w}_{test}, \mathbf{w}_{target}) = \cos(\theta) = \frac{\mathbf{w}_{test} \cdot \mathbf{w}_{target}}{\|\mathbf{w}_{test}\| \|\mathbf{w}_{target}\|} \in [-1, 1]$$

PLDA

- **PLDA (Probabilistic Linear Discriminant Analysis)**: It follows modeling assumptions similar to JFA

An i-vector $\mathbf{w}_{s,h}$ can be decomposed as:

$$\mathbf{w}_{s,h} = \mathbf{w}_0 + \mathbf{\Phi}\boldsymbol{\beta}_s + \mathbf{\Gamma}\boldsymbol{\alpha}_h + \boldsymbol{\varepsilon}_{s,h}$$

1. $\mathbf{w}_0$ is an average, *speaker-independent* i-vector
2. $\mathbf{\Phi}$ and $\mathbf{\Gamma}$ are low-rank matrices that characterize speaker and channel subspaces
3. $(\boldsymbol{\beta}_s, \boldsymbol{\alpha}_h) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ are speaker and channel factors
4. $\boldsymbol{\varepsilon}_{s,h}$ is a residual vector

- If $\mathbf{w}_{s,h} \leftarrow \mathbf{w}_{s,h}/\|\mathbf{w}_{s,h}\|_2$, $\mathbf{w}_{s,h} \sim \mathcal{N}$ (Gaussian PLDA model)
- Using a full-covariance model for $\boldsymbol{\varepsilon}_{s,h} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_\varepsilon)$ allows us to simplify the PLDA model:

$$\mathbf{w}_{s,h} = \mathbf{w}_0 + \mathbf{\Phi}\boldsymbol{\beta}_s + \boldsymbol{\varepsilon}_{s,h}$$

PLDA

- We extract $\mathbf{w}_{test}$ from a sample for verification and claim the identity associated with $\mathbf{w}_{target}$

Hypothesis contrast:

1. H_0: $\mathbf{w}_{test}$ and $\mathbf{w}_{target}$ come from the same speaker
2. H_1: $\mathbf{w}_{test}$ and $\mathbf{w}_{target}$ come from different speakers

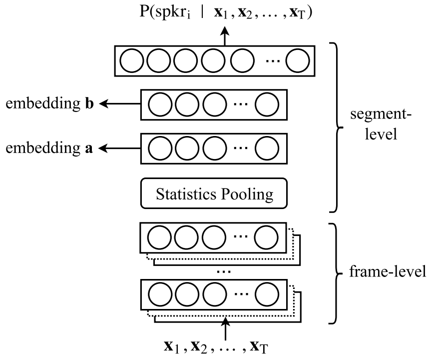
$$\mathrm{LLR}(\mathbf{w}_{test}, \mathbf{w}_{target}) = \log\left(\frac{P(\mathbf{w}_{test}, \mathbf{w}_{target}|H_0)}{P(\mathbf{w}_{test}|H_1)P(\mathbf{w}_{target}|H_1)}\right)$$

- $\mathrm{LLR}(\mathbf{w}_{test}, \mathbf{w}_{target})$ can be approximated as a function of $\mathbf{\Phi}$ and $\mathbf{\Sigma}_\varepsilon$
- If $\mathrm{LLR}(\mathbf{w}_{test}, \mathbf{w}_{target}) \geq \tau$, we accept that $\mathbf{w}_{test}$ and $\mathbf{w}_{target}$ come from the same speaker

5th Generation: Neural Networks (2018)

- While there were multiple neural network-based proposals, it was the **x-vectors** that broke the mold

TDNN (Time-Delay Neural Network)



[Snyder17] D. Snyder *et al.*, "Deep Neural Network Embeddings for Text-Independent Speaker Verification," in Proc. of Interspeech 2017

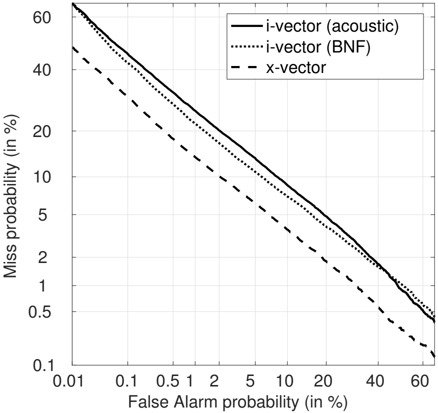
Layer	Layer context	Total context	Input x output
frame1	$[t-2, t+2]$	5	120x512
frame2	$\{t-2, t, t+2\}$	9	1536x512
frame3	$\{t-3, t, t+3\}$	15	1536x512
frame4	$\{t\}$	15	512x512
frame5	$\{t\}$	15	512x1500
stats pooling	$[0, T)$	T	1500Tx3000
segment6	$\{0\}$	T	3000x512
segment7	$\{0\}$	T	512x512
softmax	$\{0\}$	T	512xN

[Snyder18] D. Snyder *et al.*, "X-Vectors: Robust DNN Embeddings for Speaker Recognition," in Proc. of ICASSP 2018

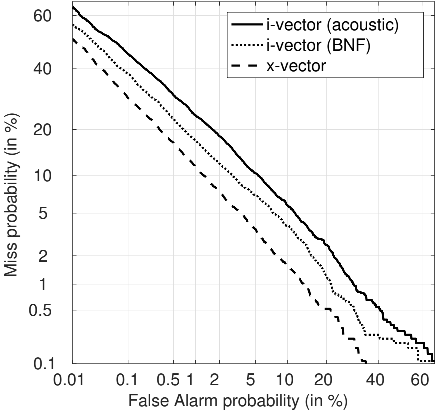
- x-vectors: Output of the *segment6* layer (from the Mel spectrogram)
- The comparison of x-vectors is performed using PLDA
- Training data augmentation is key!

5th Generation: Neural Networks (2018)

DET curve for the NIST SRE16 Cantonese set



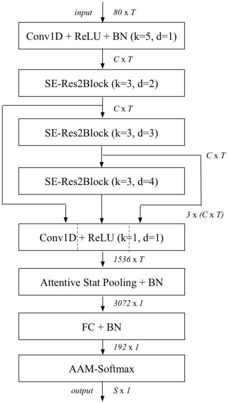
DET curve for “Speakers In The Wild”



[Snyder18] D. Snyder *et al.*, “X-Vectors: Robust DNN Embeddings for Speaker Recognition,” in Proc. of ICASSP 2018

5th Generation: Neural Networks (2018)

- **ECAPA-TDNN**: Enhanced TDNN for speaker embedding extraction

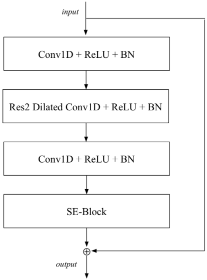


[Desplanques20] B. Desplanques *et al.*, "ECAPA-TDNN: Emphasized Channel Attention, Propagation and Aggregation in TDNN Based Speaker Verification," in Proc. of Interspeech 2020

Attentive Stat Pooling

$$e_{t,c} = \mathbf{v}_c^\top f(\mathbf{W}\mathbf{h}_t + \mathbf{b}) + k_c$$

$$\alpha_{t,c} = \frac{\exp\left(e_{t,c}\right)}{\sum_\tau^T \exp\left(e_{\tau,c}\right)}$$

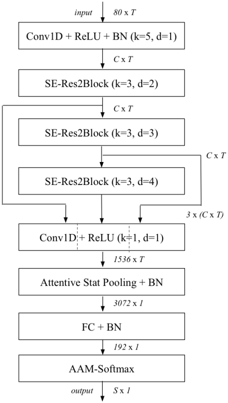


$$\tilde{\mu}_c = \sum_t^T \alpha_{t,c} h_{t,c} \qquad \tilde{\sigma}_c = \sqrt{\sum_t^T \alpha_{t,c} h_{t,c}^2 - \tilde{\mu}_c^2}$$

$$\tilde{\boldsymbol{\mu}} = (\tilde{\mu}_1, \ldots, \tilde{\mu}_c, \ldots, \tilde{\mu}_C) \qquad \tilde{\boldsymbol{\sigma}} = (\tilde{\sigma}_1, \ldots, \tilde{\sigma}_c, \ldots, \tilde{\sigma}_C)$$

5th Generation: Neural Networks (2018)

- **ECAPA-TDNN**: Enhanced TDNN for speaker embedding extraction



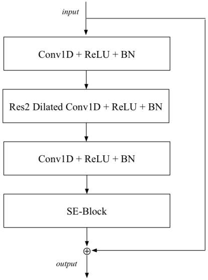
[Desplanques20] B. Desplanques *et al.*, "ECAPA-TDNN: Emphasized Channel Attention, Propagation and Aggregation in TDNN Based Speaker Verification," in Proc. of Interspeech 2020

SE (Squeeze-and-Excitation)-Block

Squeeze: $\mathbf{z} = \frac{1}{T}\sum_{t}^{T} \mathbf{h}_t$

Excitation: $\mathbf{s} = \sigma(\mathbf{W}_2 f(\mathbf{W}_1 \mathbf{z} + \mathbf{b}_1) + \mathbf{b}_2)$

$$\tilde{\mathbf{h}}_c = s_c \mathbf{h}_c \quad (s_c \in [0,1])$$



- Multi-layer feature aggregation

5th Generation: Neural Networks (2018)

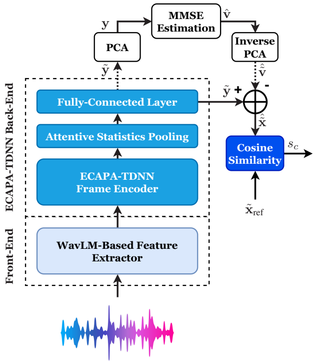
- The comparison of speaker embeddings is carried out by means of cosine similarity:

Architecture	# Params	VoxCeleb1		VoxCeleb1-E		VoxCeleb1-H		VoxSRC19
		EER(%)	MinDCF	EER(%)	MinDCF	EER(%)	MinDCF	EER(%)
E-TDNN	6.8M	1.49	0.1604	1.61	0.1712	2.69	0.2419	1.81
E-TDNN (large)	20.4M	1.26	0.1399	1.37	0.1487	2.35	0.2153	1.61
ResNet18	13.8M	1.47	0.1772	1.60	0.1789	2.88	0.2672	1.97
ResNet34	23.9M	1.19	0.1592	1.33	0.1560	2.46	0.2288	1.57
ECAPA-TDNN (C=512)	6.2M	1.01	0.1274	1.24	0.1418	2.32	0.2181	1.32
ECAPA-TDNN (C=1024)	14.7M	**0.87**	**0.1066**	**1.12**	**0.1318**	**2.12**	**0.2101**	**1.22**

[Desplanques20] B. Desplanques *et al.*, "ECAPA-TDNN: Emphasized Channel Attention, Propagation and Aggregation in TDNN Based Speaker Verification," in Proc. of Interspeech 2020

Embedding Compensation

- Need for compensation of embeddings from **non-neutral phonation speech** (e.g., shouted and whispered)



[Espejo23] I. López-Espejo *et al.*, "Improved Vocal Effort Transfer Vector Estimation for Vocal Effort-Robust Speaker Verification," in Proc. of MLSP 2023

$\tilde{\mathbf{x}} \in \mathbb{R}^D$: Normal embedding
$\tilde{\mathbf{y}} \in \mathbb{R}^D$: Non-neutral phonation embedding
$\tilde{\mathbf{v}} \in \mathbb{R}^D$: Vocal effort transfer vector

$$\tilde{\mathbf{y}} = \tilde{\mathbf{x}} + \tilde{\mathbf{v}} \Rightarrow \hat{\tilde{\mathbf{x}}} = \tilde{\mathbf{y}} - \hat{\tilde{\mathbf{v}}}$$

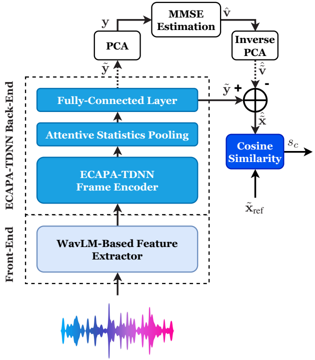
- $\mathbf{v} = \mathbf{W}_L^\top \tilde{\mathbf{v}}$ and $\mathbf{y} = \mathbf{W}_L^\top \tilde{\mathbf{y}}$, where $\mathbf{W}_L \in \mathbb{R}^{D\times L}$, $L \ll D$, is a PCA transform matrix
- $p(\mathbf{z} = (\mathbf{v}, \mathbf{y}) \in \mathbb{R}^{2L})$ is modeled by a GMM

$$\hat{\tilde{\mathbf{x}}} = \tilde{\mathbf{y}} - \underbrace{\mathbf{W}_L \hat{\mathbf{v}}}_{\hat{\tilde{\mathbf{v}}}}$$

$$\hat{\mathbf{v}} = \mathbb{E}[\mathbf{v}|\mathbf{y}] \text{ (MMSE estimation)}$$

Embedding Compensation

- Need for compensation of embeddings from **non-neutral phonation speech** (e.g., shouted and whispered)



[Espejo23] I. López-Espejo *et al.*, "Improved Vocal Effort Transfer Vector Estimation for Vocal Effort-Robust Speaker Verification," in Proc. of MLSP 2023

- The ECAPA-TDNN is trained on an augmented version of VoxCeleb2
- The evaluation metric is **EER (%)** (*the lower, the better*)

Shouted and normal speech:

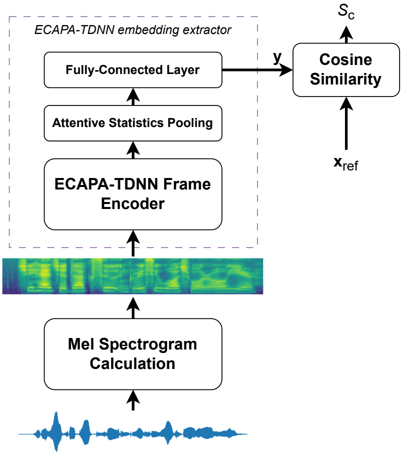
Condition	E-T+MFCC	E-T+WavLM	MEMLIN	MEMLIN+PCA	MMSE_x	MMSE_v
A_S-A_S	19.96	17.11	15.62	31.50	28.72	**15.22**
N_S-N_S	9.73	**7.25**	**7.25**	**7.25**	**7.25**	**7.25**
S-S	11.58	9.94	10.44	27.46	25.53	**5.91**
N_S-S	25.28	21.76	20.74	41.00	35.56	**17.74**

Whispered and normal speech:

Condition	E-T+MFCC	E-T+WavLM	MEMLIN	MEMLIN+PCA	MMSE_x	MMSE_v
A_W-A_W	16.54	11.24	**8.25**	31.87	23.95	8.27
N_W-N_W	1.21	**0.62**	**0.62**	**0.62**	**0.62**	**0.62**
W-W	4.38	5.26	4.00	19.31	19.77	**2.87**
N_W-W	12.81	9.81	11.47	44.38	30.59	**8.86**

Overview

Implementation of a Speaker Verification System



Implementation of a speaker verification system:

- Mel spectrogram computation
- Extraction of embeddings $\mathbf{y}, \mathbf{x}_{ref} \in \mathbb{R}^{192}$ based on an ECAPA-TDNN
- Comparison of embeddings by means of cosine similarity (if $\mathcal{S}_c \geq \tau$, $\mathbf{y}$ and $\mathbf{x}_{ref}$ come from the same speaker)

🤗 https://huggingface.co/yangwang825/ecapa-tdnn-vox2

Implementation of a Speaker Verification System

Installing Anaconda and Spyder:

1. Go to `https://www.anaconda.com/download`, download, and install `Anaconda Distribution`
2. Run `anaconda-navigator`
3. In *Environments*, create the work environment `spkVerif`
4. Select the new environment, see the *Not Installed* packages, and tick off and install Spyder
5. In *Home*, select the environment `spkVerif` and launch Spyder
6. In Spyder, go to *Tools*→*Preferences*→*iPython Console*→*Graphics*→*Backend* and choose *Automatic* as the display mode for the graphics

Implementation of a Speaker Verification System

- **We need to install some work libraries**:
  - `PyTorch`: Deep learning model construction
  - `SpeechBrain`: Development of speech technology, audio technology, etc.
  - `NumPy`: Scientific computing
  - `Sounddevice`: Sound recording and playback
  - `Matplotlib`: Generation of visualizations

1. Run `Anaconda Prompt` and activate your work environment by the command `conda activate spkVerif`
2. Install the above modules by means of `pip`: `pip install torch speechbrain numpy sounddevice matplotlib`

Implementation of a Speaker Verification System

- We create a Python script named `spkverif_lib.py`
- We **import the modules** we will need:

```
import torch
from speechbrain.pretrained.interfaces import Pretrained
import numpy as np
```

Implementation of a Speaker Verification System

- We create the **class that defines our embedding extractor based on Mel spectra**:

```
class EmbeddingExtractor(Pretrained):

    # Modules needed.
    MODULES_NEEDED = [
        "compute_features",
        "mean_var_norm",
        "embedding_model"
    ]

    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)

    # Method that performs the embedding extraction itself.
    # wav: Audio samples.
    def embedding_extractor(self, wav):

        wav = torch.tensor(wav)  # We convert the NumPy array into a tensor.

        wav = wav.unsqueeze(0)  # We add a dimension to the left of the sound segment.

        # Mel feature extraction.
        feats = self.mods.compute_features(wav)
        feats = self.mods.mean_var_norm(feats, torch.ones(1))  # Mean and variance normaliza-
tion.

        # Embedding extraction from Mel features.
        embedding = self.mods.embedding_model(feats, torch.ones(1))

        # We convert the embedding into a NumPy array.
        embedding = embedding.numpy()
        embedding = embedding[0,0,:]

        return embedding
```

Implementation of a Speaker Verification System

- We include a method to **calculate the cosine similarity** to determine whether or not two given embeddings come from the same speaker:

```
def cosineDist(x, y):

    # We calculate the vector norms.
    nx = np.sqrt(np.sum(x**2))
    ny = np.sqrt(np.sum(y**2))

    # We normalize the vectors.
    x /= nx
    y /= ny

    # We calculate cosine similarity.
    Sc = np.dot(x,y)

    return Sc
```

Implementation of a Speaker Verification System

- We will implement a very basic code with which **we can record two voice samples and determine whether or not they come from the same speaker**
- We create a new Python script named `Demo_Live.py` and import the modules we will require:

```
import spkverif_lib
import sounddevice as sd
import matplotlib.pyplot as plt
import numpy as np
```

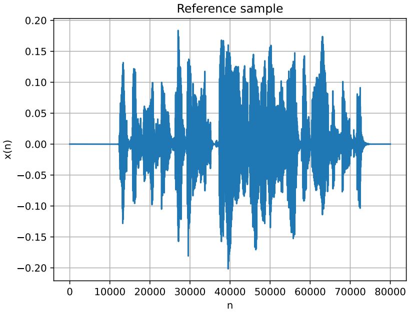
Implementation of a Speaker Verification System

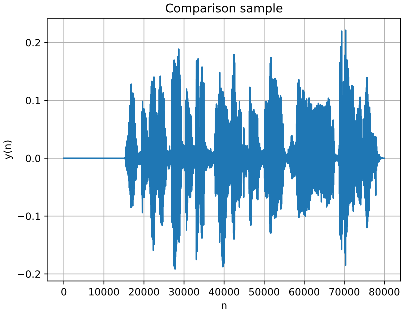
- **We record two voice samples** with a duration of 5 seconds each at a 16 kHz sampling rate and **plot them**:

```
fs = 16000  # Working sample rate in Hz: 16 kHz.
seconds = 5  # Duration of sound samples in seconds.
# ------------------------------------------------------------------------- #
input('Press any key to start recording the reference sample...')
print('Recording...')
samp1 = sd.rec(int(seconds * fs), samplerate=fs, channels=1)
sd.wait()
print('Reference sample recording complete!')
# ------------------------------------------------------------------------- #
input('Press any key to start recording the comparison sample...')
print('Recording...')
samp2 = sd.rec(int(seconds * fs), samplerate=fs, channels=1)
sd.wait()
print('Comparison sample recording complete!')
# ------------------------------------------------------------------------- #
plt.figure()
plt.plot(samp1)
plt.grid(True)
plt.xlabel('n')
plt.ylabel('x(n)')
plt.title('Reference sample')
plt.figure()
plt.plot(samp2)
plt.grid(True)
plt.xlabel('n')
plt.ylabel('y(n)')
plt.title('Comparison sample')
```

Implementation of a Speaker Verification System

- Example of two voice samples recorded by me using the present implementation
- From the left sample we will extract $\mathbf{x}_{ref}$, and, from the right one, $\mathbf{y}$:





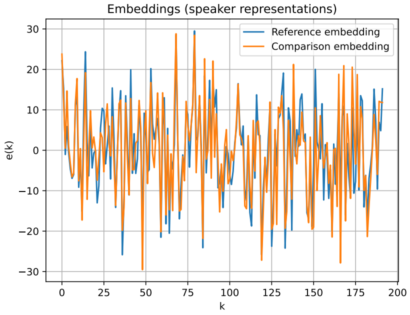
Implementation of a Speaker Verification System

- **We instantiate**, *from a repository*, **our embedding extractor** based on an ECAPA-TDNN model pre-trained using SpeechBrain
- We extract the embeddings $\mathbf{x}_{ref}$ and $\mathbf{y}$, and plot them:

```
# We instantiate the embedding extractor.
emb_extractor = spkverif_lib.EmbeddingExtractor.from_hparams(
    source='yangwang825/ecapa-tdnn-vox2'
)
embedding_1 = emb_extractor.embedding_extractor(samp1[:,0])  # Reference embedding.
embedding_2 = emb_extractor.embedding_extractor(samp2[:,0])  # Comparison embedding.
plt.figure()
plt.plot(embedding_1)
plt.plot(embedding_2)
plt.legend(['Reference embedding', 'Comparison embedding'])
plt.xlabel('k')
plt.ylabel('e(k)')
plt.grid(True)
plt.title('Embeddings (speaker representations)')
```

Implementation of a Speaker Verification System

- Next, we see the reference embedding, $\mathbf{x}_{ref}$, and comparison embedding, $\mathbf{y}$, corresponding to our previous example:



- Their **similarity** is evident, **consistent with** the fact **that they come** from voice samples **from the same person**

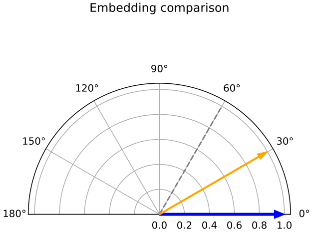
Implementation of a Speaker Verification System

- We calculate the cosine similarity $\mathcal{S}_c$ between $\mathbf{x}_{ref}$ and $\mathbf{y}$; if $\mathcal{S}_c \geq \tau = 0.5$, we state that the two voice samples come from the same speaker
- We plot the two embeddings in the polar coordinate space

```
sc = spkverif_lib.cosineDist(embedding_1, embedding_2)  # Cosine similarity.
thr = 0.5  # Decision threshold.
if sc >= thr:
   print('The two samples come from the same speaker')
else:
   print('The samples come from different speakers')
print('Cosine similarity: ' + str(sc))
# We represent the result in terms of the angle between the reference and
# comparison embeddings.
angle_ref = 0  # Reference.
angle_com = np.acos(sc)  # Comparison (arc-cosine of the cosine similarity).
arrow_length = 1.0  # Embedding length.
fig, ax = plt.subplots(subplot_kw={'projection': 'polar'})
# We only show the semicircle of interest.
ax.set_thetamin(0)
ax.set_thetamax(180)
# Reference embedding.
ax.annotate("",  # No-text annotation.
         xy=(angle_ref, arrow_length),  # Tip of the embedding.
         xytext=(0, 0),  # Origin of the embedding.
         arrowprops=dict(facecolor='blue', edgecolor='blue', width=2, headwidth=8, head-
length=10, shrink=0),
         annotation_clip=False)  # Prevent the arrow from being clipped.
# Comparison embedding.
ax.annotate("",
         xy=(angle_com, arrow_length),
         xytext=(0, 0),
         arrowprops=dict(facecolor='orange', edgecolor='orange', width=1, headwidth=6,
headlength=10, shrink=0),
         annotation_clip=False)
ax.plot([0, np.acos(thr)], [0, arrow_length], linestyle='--', color='gray')  # We also represent
the decision threshold.
ax.set_title('Embedding comparison')
ax.grid(True)
plt.show()
```

Implementation of a Speaker Verification System

- Embeddings that form an angle $\theta \leq \theta_\tau = \arccos(\tau = 0.5) = 60^\circ$ (**decision threshold**≡dashed line) come from the same speaker
- The comparison embedding $\mathbf{y}$ forms an angle $\theta = \arccos(\mathcal{S}_c)$ with the reference embedding $\mathbf{x}_{ref}$



- It is correctly determined that **the two voice samples in the example come from the same speaker**

Acknowledgment: This Topic in the Summer School has been funded by the "**Thematic Network on Applications of sound signal processing in society and digital industry**" with reference RED2022-134742-T funded by MCIN/AEI/10.13039/501100011033

Forensic Acoustics: An Introduction to Voice Identification

Iván López-Espejo, PhD

EAA SUMMER SCHOOL
Applications of Sound Signal Processing in Society and Digital Industry

iloes@ugr.es

Saturday 21st June, 2025

FORUM ACUSTICUM EURONOISE 2025
23-26 JUNE 2025

UNIVERSIDAD DE GRANADA